KING SAUD UNIVERSITY
COLLEGE OF COMPUTER & INFORMATION SCIENCES
DEPT OF COMPUTER SCIENCE

CSC311 Design and Analysis of Algorithms
Second Semester 1444
Instructor: Prof. Mohamed Maher Ben Ismail

Tutorial #2

By. 3meer

1. Consider the pseudo-code below:

ALGORITHM *MaxElement*(***A***[0..***n*** − 1])

//Determines the value of the largest element in a given array
//Input: An array ***A***[0..***n*** − 1] of real numbers
//Output: The value of the largest element in ***A*** ***maxval*** ← ***A***[0]
for *i* ← 1 to *n* − 1 do ⟶ $\sum_{i=1}^{n-1} 1 = n-1+1 = n$
  if *A*[*i*] > *maxval* ⟶ $n-1$
    maxval ← *A*[*i*]
return *maxval* $C(n) = 2n - 1$

a. What is the basic operation of this algorithm?
b. Give the best-case and worst-case time complexities of this algorithm in asymptotic notation.

b)

Worst case	Best case
$2n - 1 \leq 2n + n$	$2n - 1 \geq n$
$\leq 2n$	$n_0 = \frac{\sum_{i=0}^{k-1} \lvert a_i \rvert}{a_k - c} = \frac{1}{2-1} = 1$
$c =$ $n_0 = 1$	$c = 1$ $n_0 = 1$
$O(n)$	$\Omega(n)$

2. Consider the algorithm below and give its best-case and worst-case time complexities in asymptotic notation.

ALGORITHM *UniqueElements*(***A***[0..***n*** − 1])

//Determines whether all the elements in a given array are distinct //Input: An array ***A***[0..***n*** − 1]
//Output: Returns “true” if all the elements in ***A*** are distinct // and “false” otherwise

for $i \leftarrow 0$ to $n-2$ do → $\sum_{i=0}^{n-2} 1+1 = n-2+1+1 = n$

  for $j \leftarrow i+1$ to $n-1$ do → $\sum_{i=0}^{n-2}\left(\sum_{j=i+1}^{n-1} 1+1\right) = \sum_{i=0}^{n-2} n-i = \sum_{i=0}^{n-2} n - \sum_{i=0}^{n-2} i$

$n-1-i-1+1+1 = n-i$

$= n^2-n-\left(\frac{(n-2)(n-1)}{2}\right)$

$= n^2-n-\frac{n^2-n-2n+2}{2}$

$= n^2-n-\frac{n^2-3n+2}{2}$

$= \frac{2n^2-2n-n^2+3n-2}{2}$

$= \frac{n^2+n-2}{2}$

    if $A[i] = A[j]$ return False → $\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} n-1-i$

$= \sum_{i=0}^{n-2} n - \sum_{i=0}^{n-2} 1 - \sum_{i=0}^{n-2} i$

$= (n)(n-1) - \frac{(n-2)(n-1)}{2} - (n-1)$

$= \frac{2n(n-1)-(n-2)(n-1)-2(n-1)}{2}$

$= \frac{(n-1)(2n-n+2-2)}{2}$

$= \frac{n^2-n}{2}$

return True → 1

$f(n) = n + \frac{n^2+n-2}{2} + \frac{n^2-n}{2} + 1$

$f(n) = \frac{2n+n^2+n-2+n^2-n+2}{2} = \frac{2n^2-2n}{2}$

$f(n) = 4n^2-4n$

Worst case

$4n^2-4n \leq 4n^2+4n^2$

$\leq 8n^2$

$c = 8 \quad n_0 = 1$

$O(n^2)$

Best case

$4n^2-4n \geq c \cdot 4$

$n_0 = \frac{\sum_{i=0}^{k-1}|a_i|}{a_k - c} = \frac{4}{4-3} = 4$

$c = 3 \quad n_0 = 4$

$\Omega(1)$

KING SAUD UNIVERSITY
COLLEGE OF COMPUTER & INFORMATION SCIENCES
DEPT OF COMPUTER SCIENCE

3. Consider the algorithm below:

> **ALGORITHM** ***F(n)***
> //Computes ***n***! recursively //Input: A nonnegative integer ***n*** //Output: The value of ***n***!
> if $n = 0$ return 1
> else return $F(n-1) * n$

a. What is the algorithm's basic operation?
b. What is the resulting recursive equation?
c. Solve the equation you gave in (b).
d. What is the worst case time complexity of this algorithm?

a) Multiplication

b) $T(n) = T(n-1) + 1$

c) $= T(n) = [T(n-2)+1] + 1$

$= T(n-2) + 2$

$\vdots$ after k Substitution

$T(n-k) + k$

$= T(n-(n-1)) + (n-1)$

$= 1 + n - 1$

$\therefore T(n) = n$

Stop

$n - k = 1$

$k = n - 1$

d)

$$n \leq c\,g(n)$$
$$\leq n$$
$$c = 1 \quad n_0 = 1$$
$$\therefore O(n)$$

4. Consider the algorithm below:

> **ALGORITHM** ***Q(n)***
>
> //Input: A positive integer ***n***
>
> if $n = 1$ return 1
>
> else return $Q(n-1) + 2 * n$

a. What is the algorithm's basic operation?
b. What is the resulting recursive equation?
c. Solve the equation you gave in (b).
d. What is the worst case time complexity of this algorithm?

a) Multiplication

b) $T(n) = T(n-1) + 1$

c) $= T(n) = [T(n-2)+1] + 1$

$$= T(n-2) + 2$$

Stop

$n-k=1$

$k=n-1$

$\vdots$ after k

$\vdots$ Substitution

$$T(n-k)+k$$

$$= T(n-(n-1)) + (n-1)$$

$$= 1 + n - 1$$

$$\therefore T(n) = n$$

d)

$$n \leq c\,g(n)$$

$$\leq n$$

$c = 1 \quad n_0 = 1$

$$\therefore O(n)$$